EDGEWOOD CHEMICAL BIOLOGICAL CENTER
U.S. ARMY RESEARCH, DEVELOPMENT AND ENGINEERING COMMAND
Aberdeen Proving Ground, MD 21010-5424

ECBC-TN-067

ALGORITHM—EIGENVALUE ESTIMATION OF HYPERSPECTRAL WISHART COVARIANCE MATRICES FROM A LIMITED NUMBER OF SAMPLES

Avishai Ben-David

RESEARCH AND TECHNOLOGY DIRECTORATE

Charles E. Davidson

SCIENCE AND TECHNOLOGY CORPORATION
Belcamp, MD 21017-1427

March 2015

Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorizing documents.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE ***(DD-MM-YYYY)***
XX-03-2015

2. REPORT TYPE
Final

3. DATES COVERED ***(From - To)***
Jan 2011 – Nov 2012

4. TITLE AND SUBTITLE
Algorithm—Eigenvalue Estimation of Hyperspectral Wishart Covariance Matrices from a Limited Number of Samples

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Ben-David, Avishai (ECBC) and Davidson, Charles E. (STC)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Director, ECBC, ATTN: RDCB-DRD-P, APG, MD 21010-5424
Science and Technology Corporation (STC), 111-C Bata Blvd., Belcamp, MD 21017-1427

8. PERFORMING ORGANIZATION REPORT NUMBER
ECBC-TN-067

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Edgewood Chemical Biological Center, Aberdeen Proving Ground, MD 21010-5424

10. SPONSOR/MONITOR'S ACRONYM(S)
ECBC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT-LIMIT 200 WORDS
This MATLAB function is an algorithm designed to improve the eigenvalue estimates of Wishart-distributed covariance matrices and to recompute a revised covariance matrix from the eigenvalues. The MATLAB function is an implementation of the procedure developed and published by Avishai Ben-David and Charles E. Davidson (Eigenvalue Estimation of Hyperspectral WishartCovariance Matrices from Limited Number of Samples. *IEEE Trans. Geosci. Remote Sens.* **2012**, *50*(11), pp 4384–4396).

15. SUBJECT TERMS

Covariance matrices	Detection algorithms	Eigenvalues and Eigenfunctions	Hyperspectral sensors
Regularization	Signal processing algorithms	Stochastic processes	MATLAB function

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Renu B. Rastogi
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
U	U	U	UU	18	(410) 436-7545

Blank

PREFACE

The work described in this report was started in January 2011 and completed in November 2012.

The use of either trade or manufacturers' names in this report does not constitute an official endorsement of any commercial products. This report may not be cited for purposes of advertisement.

This report has been approved for public release.

Blank

CONTENTS

Blank

ALGORITHM—EIGENVALUE ESTIMATION OF HYPERSPECTRAL WISHART COVARIANCE MATRICES FROM A LIMITED NUMBER OF SAMPLES

1. INTRODUCTION

Estimating the eigenvalues of a covariance matrix has many applications (e.g., biometrics for physiological characteristics, hyperspectral remote sensing for detecting signals buried in noise and clutter, and medical genetics for identifying population structure). It is also fundamentally important in techniques such as principal component analysis and linear discriminant analysis, both of which are heavily used in chemometrics and in discrimination algorithms.

The estimation of all p eigenvalues of a sampled covariance matrix, when the number of samples (n) is small but $n/p>1$, is a challenging problem. In 2012, Ben-David and Davidson[*] presented a new method, with which one can improve the eigenvalue estimates of a Wishart distributed covariance matrix and an improved covariance matrix from the eigenvalues can be obtained. In this technical note, we present a MATLAB algorithm for computing the eigenvalues and an improved covariance matrix that is constructed from the eigenvalues.

[*] Ben-David, Avishai; Davidson, Charles E. Eigenvalue Estimation of Hyperspectral WishartCovariance Matrices from Limited Number of Samples. *IEEE Trans. Geosci. Remote Sens*. **2012**, *50*(11), pp 4384–4396 (DOI: 10.1109/TGRS.2012.2191415).

.

2. MATLAB CODE

```
function [C,opt]=eigenvalue_estimation(Cx,n,varargin);
% EIGENVALUE_ESTIMATION improves estimation of the eigenvalues of a covariance matrix
% This MATLAB function is an algorithm designed to improve the eigenvalue estimates of Wishart-
% distributed covariance matrices and to recompute an improved covariance matrix from the
% eigenvalues. The function is an implementation of the procedure developed and published by
% Avishai Ben-David and Charles E. Davidson, "Eigenvalue Estimation of Hyperspectral Wishart
% Covariance Matrices From Limited Number of Samples", IEEE Trans. Geosci. Remote Sens., Vol. 50,
% no. 11, pp. 4384- 4396, November 2012.
%
% Usage
%
%   [C,opt]=eigenvalue_estimation(Cx,n,varargin);
%
% Input
%
%   Cx       - p-by-p covariance matrix or p-by-1 vector of eigenvalues in
%              descending order (i.e., as returned by SVD).  If Cx is a p-by-p
%              covariance matrix it is assumed to be symmetric and positive
%              semi-definite.
%   n        - scalar, number of degrees of freedom used in computing the
%              sample covariance matrix (number of vectors minus 1)
%   varargin - optional input variables using property/value pair syntax
%              (i.e., comma-separated list syntax, see VARARGIN for more).
%              The following properties are supported:
%
%              't', scalar value indicating the transition point between large
%                 and small eigenvalues.  The default value is [] which means
%                 that either MDL or a successive linear regression method
%                 will be used to estimate it.  Valid values are in the range
%                 2:p.
%
%              'MDL', logical scalar, specifying whether minimum description
%                 length should be used for determining the transition point,
%                 't', between small and large eigevnalues.  Default is true
%                 so the MDL is used.  False will use the successive linear
%                 regression method described in the TGRS paper, Eq. (12).  In
%                 general MDL is a good method for n/p<10.
%
%              'g', function handle defining a monotonic transformation of the
%                 eigenvalues that is applied prior to estimating 't' using
%                 the successive linear regressions.  The default is the
%                 anonymous function @(x)x which is the identity transform and
%                 represents computing the regression in "linear-scale".  Use
%                 @log for "log-scale".  In the paper estimating 't' using
%                 linear-scale was better for Telops data; estimating 't'
%                 using log-scale was better for SEBASS.  The function in 'g'
%                 will only be used if 'MDL' is set to false.
%
%              'm', p-by-1 apparent multiplicity curve.  Default is [] so that
%                 'm' will be estimated from equations (8-9) from the paper.
%                 In the paper, the apparent multiplicity is called "p_i".
%
% Output
%
%   C   - p-by-p corrected covariance matrix or p-by-1 vector of corrected
%         eigenvalues
%   opt - (optional) struct containing values of the optional input arguments
%         used internally inside the function
%
% Notes
%
%   This function implements the eigenvalue correction and estimation procedure outlined in
%   Ben-David and Davidson, TGRS 50(11), 4384-4396 (2012), which is designed
%   to improve the eigenvalue estimates of Wishart-distributed covariance
%   matrices.  It lacks the simulation and evaluation functionality of the
%   full code and only implements the correction procedure on a single sampled
%   covariance matrix (or spectrum of eigenvalues).
%
```

```
%   This function accepts either a covariance matrix or a vector of
%   eigenvalues. In the former case, the output will be a corrected
%   covariance matrix recreated using an SVD decomposition, where the
%   corrected eigenvalues take the place of the sample eigenvalues (but sample
%   eigenvectors are unaltered, for lack of knowledge). In the latter case,
%   the output will simply be a vector of corrected eigenvalues.
%
%   In the TGRS paper, we show through simulation that this procedure is able
%   to "adjust" the sample eigenvalues so that they are better estimates
%   (overall) of the eigenvalues of the population covariance matrix while
%   simultaneously improving the condition number of the covariance matrix.
%   The resulting (inverse) corrected covariance matrices improved detection
%   performance of the matched filter compared to using the original (inverse)
%   sample covariance matrix, and also improved over the Effron-Morris
%   estimator of the inverse covariance (an example of the empirical-Bayes
%   class of estimators) which takes the form of diagonal loading.  The
%   "two-subset" method (which is a technique to remove bias at the expense of
%   higher variance) showed slightly improved detection results when used in
%   the matched filter, though it is less good at improving the eigenvalue
%   estimates.
%
%   EXAMPLES
%
%      %generate some artificial input data for demo purposes
%      sx=sort(exp(randn(125,1)).*2,'descend'); %eigenvalues
%      n=300;                                   %d.o.f.
%
%      %a corresponding artificial covariance matrix
%      U=orth(rand(125)); %eigenvectors
%      Cx=U*diag(sx)*U.'; %covariance matrix
%
%      %use all defaults passing sx as input (output is corrected e.v.)
%      [s,opt]=eigenvalue_estimation(sx,n);
%
%      %use all defaults passing Cx as input (output is corrected cov. matrix)
%      [C,opt]= eigenvalue_estimation(Cx,n);
%
%      %note that C = U*diag(s)*U.';
%
%      %instead of MDL use the successive regression method (in linear space)
%      [s,opt]= eigenvalue_estimation(sx,n,'MDL',false);
%
%      %instead of MDL use the successive regression method (in log space)
%      [s,opt]= eigenvalue_estimation(sx,n,'MDL',false,'g',@log);
%
%      %specify a particular value of t to use
%      [s,opt]= eigenvalue_estimation(sx,n,'t',5);
%
% Last Documented Change: 12/10/12, CED (R2010b)


if nargin<2 || isempty(n) || isempty(Cx)
    error('"Cx" and "n" must be passed.');
end

%defaults for optional inputs (stored in a structure where the field name and
%value defines the property/value pairs)
opt.t=[];     %allow user to specify a particular value of the transition point
opt.MDL=true; %use MDL for estimating t by default if 't' is empty
opt.g=@(x)x;  %use linear-scale when 'MDL' is false and 't' is empty
opt.m=[];     %allow user to specify a particular multiplicity curve

%override defaults with user inputs
opt=localParseVarargin(opt,varargin); %subfunction

%number of dimensions
p=length(Cx);

if numel(Cx)==p
    %Cx is really a vector of eigenvalues, not a covariance matrix
    covarianceFlag=false;
```

```
    sx=Cx(:); %column vector
else
    %Cx is a covariance matrix
    covarianceFlag=true;

    %compute eigenvalues
    [U,S]=svd(Cx); %V=U'
    sx=diag(S);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%step 1: shifted eigenvalues%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if isempty(opt.m) || numel(opt.m)~=p
    %multiplicity curve needs to be estimated
    opt.m=multiplicity(sx,n); %subfunction
end

%adjust the eigenvalues
k=opt.m(:)./n; %column vector
s=sx.*(1+k)./(1-k).^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%step 2: energy normalization%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if isempty(opt.t)
    %transition location between large and small eigenvaules needs to be
    %estimated (the "knee" of the scree plot)

    if opt.MDL
        %default method is to use MDL (which is reasonable for n/p<10 and for
        %eigenvalue spectra that are not "too peaked")
        opt.t=local_MDL_knee(sx,n); %subfunction
    else
        %use the successive regression method...whether the transition point
        %is computed in "linear-space" or "log-space" will depend on the
        %function g.  By default g is linear.  Log-space is achieved by
        %setting g to be @log.  Any other transformation may be specified
        %using the function handle syntax, but problems will arise if g is not
        %monotonic, and so arbitrary g is not recommended...
        opt.t=find_the_knee(opt.g(sx)); %subfunction
    end
end

%by definition t must be within the range of 2 to p (in case user passed in an
%invalid value)
opt.t=max(min(opt.t,p),2);

large=1:opt.t-1; %index for large eigenvalues
small=opt.t:p;   %index for small eigenvalues

actual1=sum(s(small));  %trace of (small) corrected eigenvalues
target1=sum(sx(small)); %desired trace for small eigenvalues

actual2=sum(s(large));  %trace of (large) corrected eigenvalues
target2=sum(sx(large)); %desired trace for large eigenvalues

s(small)=s(small).*target1./actual1;
s(large)=s(large).*target2./actual2;

%re-sort
s=sort(s,'descend');

if covarianceFlag
    %user passed in the sample covariance, pass out the corrected covariance
    %matrix using the original eigenvectors and the corrected eigenvalues
    C=U*diag(s)*U';
else
    %user passed in the eigenvalues, just pass out corrected eigenvalues
```

```
    C=s;
end

return;


function prop=localParseVarargin(prop,v);
%LOCALPARSEVARARGIN A vastly simplified local version of PARSEVARARGIN that
%   will cycle through the optional input arguments and the fields of the
%   property/value pair struct and override the default values in the struct
%   with user-set values.  If the same property is matched twice, the last one
%   passed in varargin will be used.  Matching is case-insensitive, but
%   otherwise must be exact (abbreviations won't match).  Note also that there
%   are no warnings for failed matches (failed matches are simply ignored).

   %number of elements in v (which is the cell array varargin)
   numv=length(v);

   if mod(numv,2)
       %number of elements in varargin should have been even
       error(['Improper property/value pair syntax ',...
           '(odd number of elements appearing in varargin)']);
   end

   %get fields we will test for
   propNames=fieldnames(prop);
   N=length(propNames);

   for i=1:2:numv
       %for each odd input element
       for j=1:N
           %check all property names
           if strcmpi(v{i},propNames{j})
               %assign new value if there is a match
               prop.(propNames{j})=v{i+1};
           end
       end
   end


function m=multiplicity(s,n);
%MULTIPLICTY Subfunction to estimate of the apparent multiplicity, Eq. (9)
%   from paper, using theoretical bounds on the value of the sampled
%   eigenvalue (as a function of n,p and the population eigenvalue).  We don't
%   know the population eigenvalue, so we use the sampled eigenvalue, instead.
%   This is similar to how in a GLRT the estimated parameters take the place
%   of the population parameters in the likelihood ratio (for lack of
%   knowledge).

    p=numel(s);

    %approximate bounds on the eigenvalues used in Eq. (8) that are derived
    %from the Marcenko-Pastur law
    k=p./n;        %band-to-vector ratio
    rootk=sqrt(k); %for convenience

    lim=s(:)*(1-[rootk -rootk]).^2; %[a b]

    %Eq. (9)
    for i=p:-1:1
        m(i,1)=sum(lim(i,2)<=lim(:,2) & lim(i,2)>=lim(:,1));
    end

    %limit the multiplicity curve so that it can't decrease once it has
    %reached a maximum, Eq. (10)
    [maxm,loc]=max(m);

    if m(end)~=maxm
        m(loc:end)=maxm;
    end
```

```
function t=find_the_knee(s);
%FIND_THE_KNEE Estimate the transition point between large and small
%   eigenvalues using the successive regressions from the left and right (see
%   Eq. (12) from paper).  The terminology is based on the idea of a "scree"
%   plot, where it is common to try to find the "crook" or "elbow" or "knee"
%   of the curve.

    p=numel(s);
    x=(1:p)';

    %initialize residuals
    res1(p-2,1)=0;
    res2=res1;

    for i=2:p-1
        %compute the "from the left" residual curve
        [P1,R1]=polyfit(x(1:i),s(1:i),1);
        res1(i-1)=R1.normr;
        %res1(i-1,1)=norm(polyval(P1,x)-s);

        %compute the "from the right" residual curve
        [P2,R2]=polyfit(x(i:p),s(i:p),1);
        res2(i-1)=R2.normr;
        %res2(i-1,1)=norm(polyval(P2,x)-s);
    end

    %the transition point is the location the residual curves intersect
    [t,t]=min(abs(res1-res2));
    t=t+1;


function t=local_MDL_knee(s,n);
%LOCAL_MDL_KNEE Local subfunction based on Avi Ben-David's MDL_KNEE function.
%   This function just returns the value of the transition in the scree plot
%   (none of the other outputs returned by the main function are needed).

    p=numel(s);

    MDL(p-1,1)=0; %initialize

    %for efficiency (no need to recompute inside of loop)
    logn=log(n);

    for k=1:p-1
        Es=mean(s(k+1:p));    %arithmetic mean
        Gs=geomean(s(k+1:p)); %geometric mean

        MDL(k)=-(p-k).*n.*log(Gs./Es)+0.5.*k.*(2.*p-k).*logn;
    end

    [t,t]=min(MDL);
```

3. CONCLUSIONS

We developed a MATLAB software with which to estimate all p eigenvalues of a sampled covariance matrix (Wishart distributed) when the number of samples (n) is small but $n/p>1$ (i.e., we did not address pseudo Wishart or singular matrices where $n<p$). We were mostly interested in n/p that is less than a few tens, but our method also works for large n/p: as n/p increases, the benefit of our method diminishes, but nevertheless, our method always improves over the sampled data. Our method is practical, quick (<0.4s on 3.3GHz workstation) and simple for implementation in MATLAB. With the improved eigenvalue solution we can construct an improved covariance matrix. We extensively tested our method. The method performed well for all cases including when the population eigenvalue spectrum was highly peaked.

DISTRIBUTION LIST

The following individuals and organizations were provided with one Adobe portable distribution format (pdf) electronic version of this report:

Spectroscopy Branch,
Research and Technology Directorate,
RDCB-DRI-S
Attn: A. Ben-David
E5554
avishai.ben-david.civ@mail.mil

Attn: Branch Chief
E5560
R. Vanderbeek
richard.g.vanderbeek.civ@mail.mil
5183 Blackhawk Road
Aberdeen Proving Ground, MD 21010-5424

Defense Threat Reduction Agency
Attn: Jerry Pate, Deputy, Physical S&T Division (RD-CBT), MSC 6201
(jerry.pate@dtra.mil)
Thomas Ward, Program Manager, DTRA/RD-CBD T, Rm 3226
(thomas.ward@dtra.mil)
Eric L. Moore, Chief, Advanced & Emerging Threats Division (J9-CBS), MSC 6201
(eric.moore@dtra.mil)
8725 John J. Kingman Road
Fort Belvoir, VA 22060-6201

Department of Homeland Security
DHS ORD CSAC
Attn: George Famini, Director DHS Chemical Security Analysis Center
(george.famini@scitech.dhs.gov)
Building E3401
5183 Blackhawk Road
Aberdeen Proving Ground, MD
21010-5424

G-3 History Office
U.S. Army RDECOM
Attn: Jeffrey K. Smart, Command Historian
(jeffrey.k.smart.civ@mail.mil)
Building E5027
5183 Blackhawk Road
Aberdeen Proving Ground, MD 21010-5424

ECBC Technical Library
RDCB-DRB-BL
Attn: Janett A. Stein, Hendall Inc.
(janett.a.stein.ctr@mail.mil)
Building E3330
5183 Blackhawk Road
Aberdeen Proving Ground, MD 21010-5424

Office of the Chief Counsel
AMSRD-CC
Attn: Vicki A. Upchurch, Paralegal Specialist
(vicki.a.upchurch.civ@mail.mil)
6001 Combat Drive
Aberdeen Proving Ground, MD 21005-1846

Defense Technical Information Center
Attn: DTIC OA
8725 John J. Kingman Road, Suite 0944
Fort Belvoir, VA 22060-6218

ECBC Rock Island
RDCB-DE
Attn: Nannette M. Ramsey, Associate Director of Engineering
(nannette.m.ramsey.civ@mail.mil)
Building 62/1/SW
1 Rock Island Arsenal
Rock Island, IL 61299-7390

EDGEWOOD CHEMICAL BIOLOGICAL CENTER

A U.S. Army RDECOM Laboratory